Deep Learning for Detecting Splicing Forgeries in Images

Mentor:
Mr. Ramaguru Radhakrishnan

Team:
1. Ms. Deepthi J (CB.EN.U4CYS21014)
2. Mr. Yaswanth G (CB.EN.U4CYS21089)

Problem Statement

- The widespread ***availability of image manipulation tools*** has made it easy to create ***fake images***, leading to an increase in image forgery across various fields, including government, finance, and education.

- This poses a ***serious challenge to the authenticity and accuracy of evidences*** presented in courtrooms, potentially leading to incorrect decisions.

- As a result, there is a ***pressing need for effective detection methods for image forgery*** and to verify the genuineness of digital images.

About the Project

- **Spliced images** are created by combining two or more different images to create a single image that appears to be real.
- The splicing process involves cutting and pasting, copying, moving, warping, or blending image parts. This process could be used to create fake photos for various purposes.
- Our project aims to develop DL algorithms to detect and identify splicing forgeries in image data.
- The project trains DL algorithms with authentic and tampered images to recognize spliced image characteristics.

Image Splicing

Image splicing is the process of generating forged images, defined as *cutting some part of the image and pasting it into another image* without carrying out any pre-processing in other words we can say it is the process of *concatenating two individual images to create a new image.*

Image1 **Image2** **Spliced image**

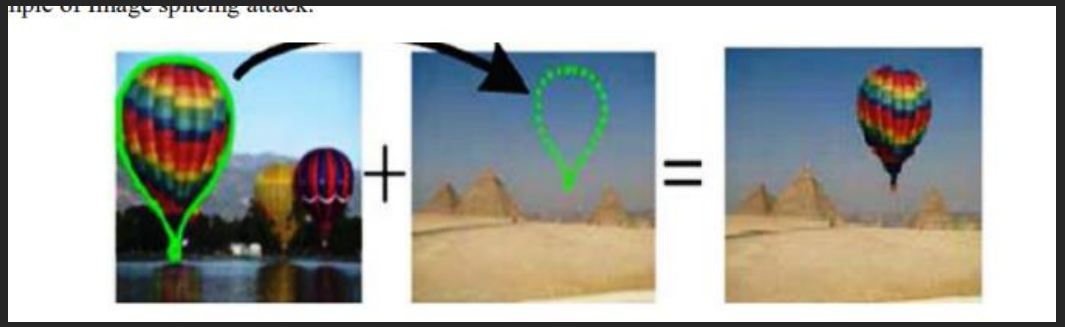


Image Morphing

Image morphing is the process of interpolating between two images in order to create something which looks like a nice blend between the two input images.

Image 1 **Image 2** **Morphed Image**

Fig. 6. Procedural transformation

Machine Learning vs Deep Learning for Splicing Detection

- **Deep Learning models usually work better** than traditional Machine Learning models because they also l**earn the feature extraction part**.
- Image splicing requires feature extractions, image classification, and object detection. In image recognition, for example, the traditional setup is to extract handcrafted features and then feed an SVM.
- But from the papers we analyzed, it is understood that DL techniques perform better than ML techniques for image analysis models (Ref Pg no. 9).
- Also, the datasets available are compactable for many DL models but mostly not for ML models.

Considering the *efficiency of Deep Learning Algorithms* in detecting the minor variations between an original and a spliced image, we chose to proceed with our project using DL methodologies.

Potential Use Cases for this Model

- **Forensic investigations:** Image Splicing Detection can be used in forensic investigations to identify fraudulent or tampered images in criminal cases.
- **News verification:** With the rise of fake news and propaganda, image splicing detection can be used to verify the authenticity of images used in news articles.
- **Social media moderation:** Image Splicing Detection can be used by social media platforms to prevent the spread of manipulated or fake images.
- **Copyright infringement:** Image Splicing Detection can be used to identify instances of copyright infringement, such as when an image is used without permission or credit.
- **Medical imaging:** Image Splicing Detection can be used in medical imaging to detect manipulated images, which can impact patient diagnoses and treatment plans.

Analysis Methodology

PAPER – 1

Fake Image Detection in Twitter Using Flood Fill Algorithm and Deep Neural Networks [2022]

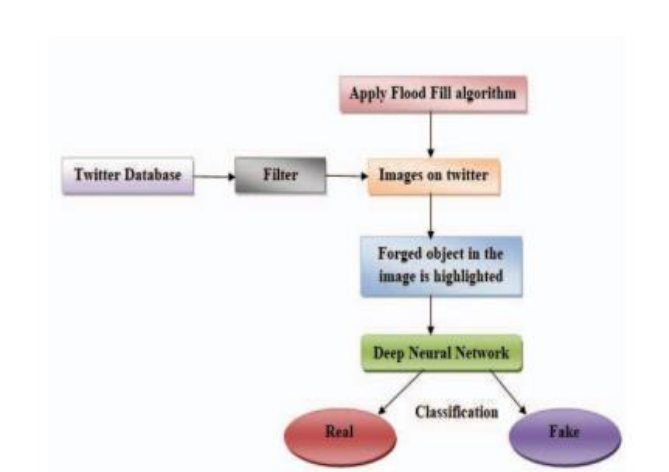


Fig. 3: Flow Diagram of the proposed framework

- ✓ In this paper, we are trying to utilize the Flood Fill algorithm to highlight the forged object in the image, and Deep Learning based solution is proposed to detect whether the image is fake or real.

- ✓ The proposed model uses Flood Fill Algorithm and Deep Learning because:

  - The Flood Fill algorithm is used to segment the image into connected regions based on their color or intensity values. Then, the characteristics of these regions, such as texture, edge features, and color distribution, can be analyzed to identify regions that may have been spliced into the image.
  - DL excels in pattern recognition. It recognizes the objects in images excellently because it uses three or even more layers of ANN, each of which is responsible for extracting one more feature from the image.

Performance Analysis

PAPER – 1

Fake Image Detection in Twitter using Flood Fill Algorithm and Deep Neural Networks [2022]

- ✓ The performance of the proposed framework is compared with the conventional models namely, Naïve Bayes, SVM, and Random Forest.
- ✓ The entire dataset gets split into 100, 200, 400, 600, and 800 samples.
- ✓ Accuracy is computed for every sample by applying the proposed framework as well as conventional models.
- ✓ The experimental results illustrated below show that the proposed framework outperforms by achieving 96% accuracy.

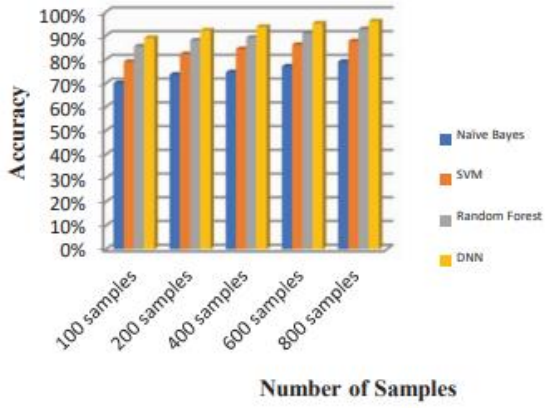


Fig.5: Accuracy Comparisons of proposed model with the existing models

- **The below table is the Precision comparisons of the proposed model(DNN) with existing models.**

	100 samples	200 samples	400 samples	600 samples	800 samples
Naïve Bayes	0.65	0.673	0.72	0.749	0.78
SVM	0.727	0.74	0.774	0.796	0.821
Random Forest	0.813	0.83	0.857	0.87	0.889
DNN	0.819	0.85	0.872	0.907	0.945

- **The below table is the Recall comparison of the proposed model(DNN) with existing models.**

	100 samples	200 samples	400 samples	600 samples	800 samples
Naïve Bayes	0.73	0.755	0.771	0.789	0.812
SVM	0.781	0.792	0.81	0.823	0.854
Random Forest	0.85	0.878	0.892	0.901	0.929
DNN	0.873	0.90	0.923	0.948	0.986

Source: Fake Image Detection in Twitter using Flood Fill Algorithm and Deep Neural Networks

Analysis Methodology

PAPER – 2

Image splicing detection using mask – RCNN [2020]

- ✓ Here ResNet-conv(convolution) is used as a backbone architecture for Deep Learning.
- ✓ ResNet-conv is obtained by replacing feature pyramid network in ResNet-FPN (Feature Pyramid Network) with a set of new convolutional layers.
- ✓ This new backbone is used to train the mask-RCNN.
- ✓ Mask-RCNN is used with ResNet model to extract the initial feature. Using Stochastic gradient descent (SGD) optimization with a momentum of 0.99 and weight decay of 0.001.
- ✓ The initial learning rate is 10%.
- ✓ The analysis is made using the dataset COCO dataset. The dataset contains 80K training images and 40K validation images. The labels are changed with a new binary mask.

Analysis Methodology

PAPER – 3

Multiple Image Splicing Dataset (MISD) [2021]

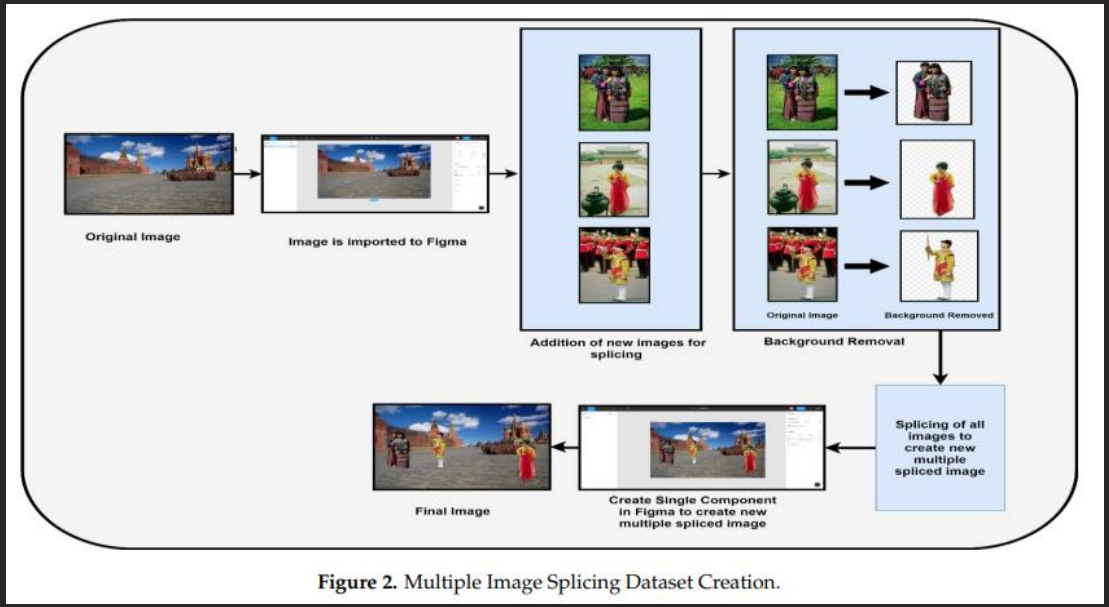


Figure 2. Multiple Image Splicing Dataset Creation.

- ✓ This research paper mainly focuses on identifying all the publicly available datasets and various methodologies that can be used for the detection of Image Splicing forgeries to increase the credibility standards of various proposed methodologies.
- ✓ The paper also highlights the process of constructing a **Multiple Image Splicing Dataset** containing high-quantity, annotated, realistic multiple spliced images.
- ✓ MISD contains 618 authentic and 300 realistic multiple spliced images by applying various processing operations such as rotating and scaling. This dataset can be used for building deep learning models for the detection of spiced images.

Table 2. Image Splicing Datasets.

Sr. No	Name of Dataset Used in Image Splicing	Total Number of Images including Authentic and Spliced Images	Dimension of Image in Pixel	Image Format	Ground Truth Mask
1	**Columbia Gray** [12]	1845 image blocks (Authentic—933; Spliced—912)	128 × 128	BMP	Not Available
2	**CUISDE** [13]	363 (Authentic—183; Spliced—180)	757 × 568 to 1152 × 768	TIFF	Available
3	**CASIA 1.0** [11]	1725 (Authentic—800; Spliced—925)	384 × 256	JPG and TIFF	Not Available
4	**CASIA 2.0** [11]	12,614 (Authentic—7491; Spliced—1849, and remaining images are of type copy move)	320 × 240 and 800 × 600	JPG and TIFF	Not Available
5	**DSO-1** [14]	200 (Authentic—100; Spliced—100)	2048 × 1536 and 1536 × 2048	JPG and TIFF	Available
6	**DSI-1** [14]	100 (Authentic—25; Splice—25)	Different sizes	PNG	Available
7	**WildWeb** [15]	10,666 (Authentic—100; Spliced—9666)	122 × 120 to 2560 × 1600	PNG	Available
8	**AbhAS** [16]	93 (Authentic—45; Spliced—48)	278 × 181 to 3216 × 4288)	JPG	Available

Table 3. Deep Learning Techniques for image splicing forgery detection.

Paper Reference	Deep Learning Technique Used for Image Splicing Forgery Detection	Dataset Used for the Experiments
[19]	CNN	CASIA V1.0 [11], CASIA V2.0 [11], Columbia Gray [12]
[20]	Deep Neural Network	CUISDE [13]
[21]	Auto Encoder Decoder, LSTM	NIST'16, IEEE Forensics Challenge Dataset, MS-COCO [22]
[23]	Mask R-CNN	Columbia Gray [12]
[24]	Mask R-CNN with backbone network as ResNet-conv	Computer-generated dataset where forged images have been generated using COCO [22] and a set of objects with transparent backgrounds where 80,000 images are used for training and 40,000 for validation. The image size is 480 × 640 pixels.
[25]	CNN	CASIA V1.0 [11], CASIA V2.0 [11]
[26]	CNN, Dense classifier network	CASIA V2.0 [11]

Data source

DEFACTO – SPLICING

- ✓ The dataset contains about 105000 splicing forgeries that are available under the splicing directory.
- ✓ Each splicing is accompanied by two binary masks.
- ✓ One under the probe_mask subdirectory, indicates the location of the forgery and one under the donor_mask indicates the location of the source.
- ✓ The external image can be found in the JSON file under the graph subdirectory.

Original Image 1

Original Image 2

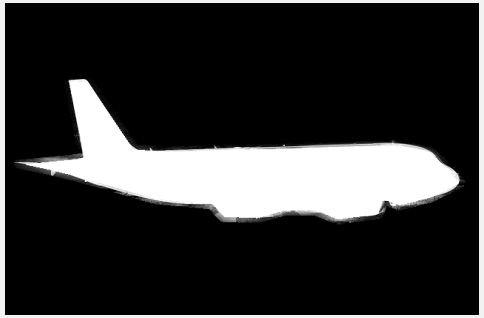
Spliced Image

Dataset Analysis

JSON Data:


```
[
    {
        "State": 0,
        "Name": "Ground Truth",
        "Property": {
            "Path":
"splicing_airplane/before/0_000000195755.jpg"
        }
    },
    {
        "State": 1,
        "Name": "Splicing",
        "Property": {
            "External image":
"MSCOCO/images/train2017/000000421322.jpg",
            "Mask":
"splicing_airplane/donor_mask/0_000000195755.tif",
            "Translation": "(573,75)",
            "Scale": "(0.26,0.26)",
            "Rotation": 0,
            "Mask feathering": 3,
            "Color Transfer": false
        }
    }
]
```


Donar Mask



A donor mask is a mask used to select a portion of an image and paste it onto another image.

Probe Mask

A probe mask is a mask used to identify spliced regions in an image by analyzing its statistical properties.

Spliced Image

A spliced image is created by copying and pasting a portion of one image onto another, resulting in an image with parts of two different images.

Masked

Spliced [translated to (573,45); Scaled down by 26%; 0^0 rotated]

Codes we analyzed from the internet

Dropped because the GUI crashed in the middle testing.

https://github.com/HXM14/Image-Forgery-Detection-using-Deep-learning

Dropped because the code wasn't working even after following all the descriptive steps and downloading all the required python models.

https://github.com/shauryagoel/Image-Splice-Detection

Code that we are using:

https://www.kaggle.com/code/alerium/defacto-test

Self-Curated Codes

✓ ***Code Version 1:***

Based on **blockiness** and **edge scores**, this method takes a picture and determines if it is spliced or not.
Link - GitHub link

✓ ***Code Version 2:***

Based on **blockiness, lighting score, texture score and edge scores**, this method takes a picture and determines if it is spliced or not.
Link - GitHub link

✓ ***Code Version 3:***

Based on **blockiness, lighting score, texture score, color score, noise score, chromatic aberration score, compression artifacts score, gradient analysis score, quantization tables score, fourier analysis score , and edge scores**, this method takes a picture and determines if it is spliced or not.
Link - GitHub link

Final version of the code for this DL - Model

Code: Deep Learning for Detecting Splicing Forgeries in Images

Link - GitHub link

UNET – Network Architecture

- ✓ UNET is a **U-shaped encoder-decoder network architecture**, which consists of four encoder blocks and four decoder blocks that are connected via a bridge.
- ✓ The encoder network (contracting path) half the spatial dimensions and double the number of filters (feature channels) at each encoder block.
- ✓ Likewise, the decoder network doubles the spatial dimensions and half the number of feature channels.
- ✓ Our model is using a **Weighted-UNET architecture**. Architecture is as follows (Next Slide)

W-UNET Architecture

Encoder Pathway:

Input Image
|
Convolutional Layers
|
ReLU Activation
|
Max Pooling
|
Convolutional Layers
|
ReLU Activation
|
Max Pooling
|
... (repeated down-sampling blocks)

Bridge:

Convolutional Layers

Decoder Pathway:

... (repeated up-sampling blocks)
|
Concatenation with corresponding encoder block
|
Convolutional Layers
|
ReLU Activation
|
Up-sampling
|
... (repeated up-sampling blocks)

Weighted Connections:

Weighted Connections between Encoder and Decoder blocks

Output Layer:

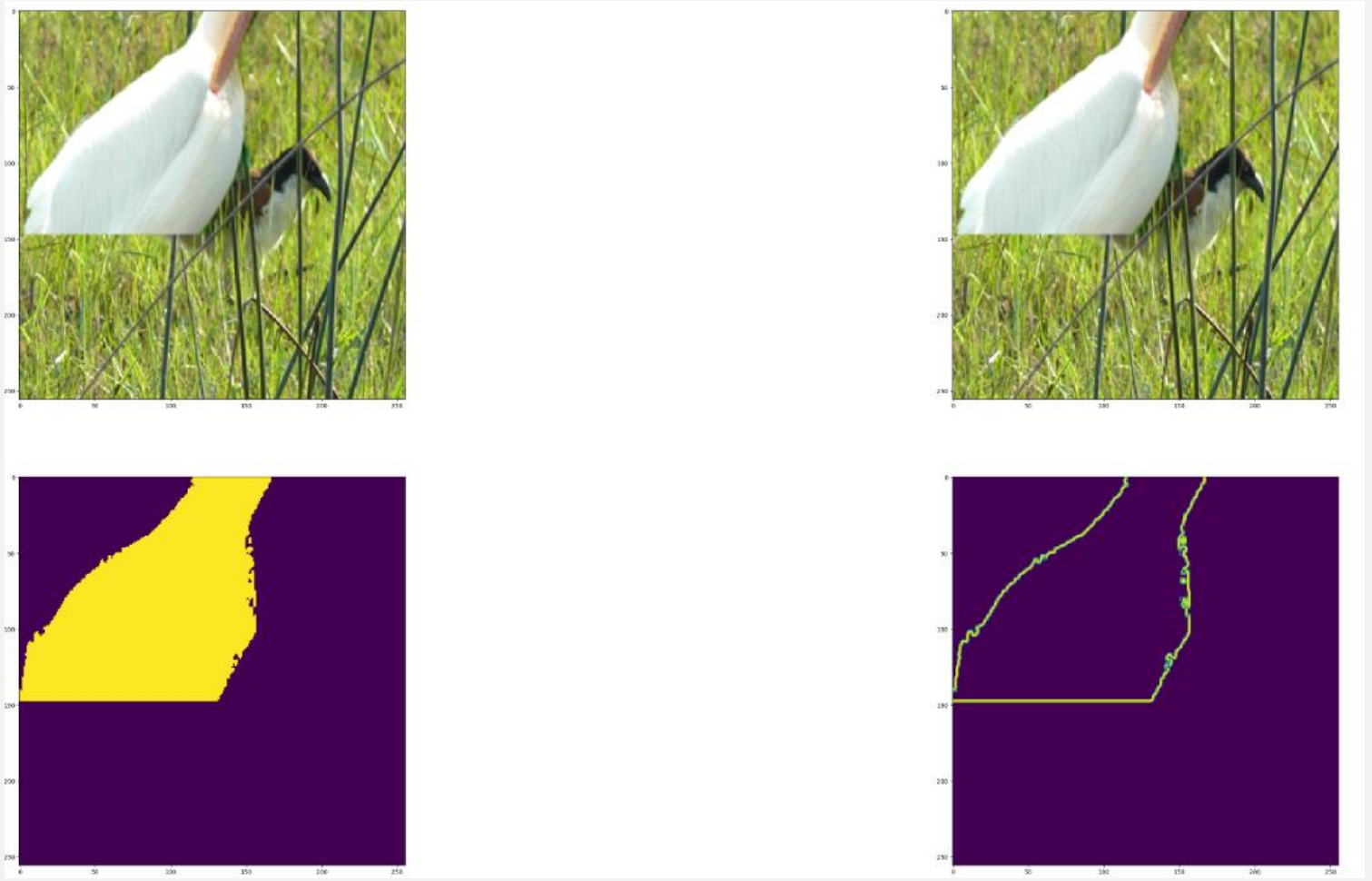
Convolutional Layer
|
Activation Function
(e.g., Sigmoid or Softmax)
|
Segmentation Map

Pre – Processed Images

Model Summary

Layer (type)	Output Shape	Param #
BayarConv2dandnoise-1	[-1, 3, 128, 128]	0
Conv2d-2	[-1, 64, 128, 128]	1,792
BatchNorm2d-3	[-1, 64, 128, 128]	128
ReLU-4	[-1, 64, 128, 128]	0
Conv2d-5	[-1, 64, 128, 128]	36,928
BatchNorm2d-6	[-1, 64, 128, 128]	128
ReLU-7	[-1, 64, 128, 128]	0
VGG-8	[-1, 64, 128, 128]	0
Softmax2d-9	[-1, 64, 64, 64]	0
GDWT-10	[-1, 64, 64, 64]	0
Conv2d-11	[-1, 64, 64, 64]	36,864
ReLU-12	[-1, 64, 64, 64]	0
BatchNorm2d-13	[-1, 64, 64, 64]	128
ConvReluBatch-14	[-1, 64, 64, 64]	0
Conv2d-15	[-1, 64, 64, 64]	73,728
ReLU-16	[-1, 64, 64, 64]	0
BatchNorm2d-17	[-1, 64, 64, 64]	128
ConvReluBatch-18	[-1, 64, 64, 64]	0
Conv2d-19	[-1, 64, 128, 128]	1,792
BatchNorm2d-20	[-1, 64, 128, 128]	128
ReLU-21	[-1, 64, 128, 128]	0
Conv2d-22	[-1, 64, 128, 128]	36,928
BatchNorm2d-23	[-1, 64, 128, 128]	128
ReLU-24	[-1, 64, 128, 128]	0
VGG-25	[-1, 64, 128, 128]	0
Softmax2d-26	[-1, 64, 64, 64]	0
GDWT-27	[-1, 64, 64, 64]	0
Conv2d-28	[-1, 64, 64, 64]	36,864
ReLU-29	[-1, 64, 64, 64]	0

Output



Challenges encountered

1. Our code required **GPU for parallel processing of multiple images** which gives better efficiency. The GPU can be instantiated by CUDA. CUDA is a parallel computing platform and programming model for general computing on GPUs. With CUDA, developers can dramatically speed up computing applications by harnessing the power of GPUs. But we couldn't activate CUDA. The reason is our graphic cards and drivers don't support CUDA.

2. We **couldn't find the algorithm used to generate the donor mask, probe mask and the Json data** due to which we weren't able to explore the edge cases of our model.

3. We c**ouldn't tap the convolution layers to show a flow diagram** of an image passing through the CNN because a particular variable required a huge amount of storage which isn't currently feasible **due to lack of resources**. The lab machines which have a RAM of 8GB took 20hr approx. to finish one epoch on being trained with just 450 images. Our laptop with a RAM of 16GB takes 45mins to finish one epoch with the same amount of training data. Hence, we didn't research further in the aspect of hardware as it wasn't required for this submission.

Conclusion

- ✓ This Deep Learning Model, used to Detect Splicing Forgeries in Image Data, is **developed using the Weighted - UNET Architecture** containing the **Encoder Pathway, Bridge, Decoder Pathway, Weighted Connections, and Output Layer** (*Ref*: slide no. 22).
- ✓ This model is **trained using 450 images** with an **accuracy of 84.6%.**
- ✓ After passing through **292 layers of the CNN**, our model will be able to **detect the edges of a spliced object**. Inputs for detection are taken from the preprocessing steps, probe mask, donor mask, and the Json data.
- ✓ Please refer to slides no. 24, 23, and 25 for the model summary, preprocessed images in batches and the output, respectively.

Annexure

Pytorch functions used in the model

Total number of layers: 292

1. **nn.Module** is base class that provides parameter management, forward propagation, and Layer encapsulation.
2. **nn.Sequential** is responsible for chain-linking the layers and redirecting the inputs and outputs into layers accordingly.
3. **nn.SoftMax2d** is used to apply SoftMax over features to each spatial location.
4. **nn.ReLU** applies the rectified linear unit function element-wise: ReLU(x)= (x)+ =max(0,x) .
5. **nn.BatchNorm2D** applies Batch Normalization over 4D input (input is an image tensor i.e., image data with in and out channels included).
6. **nn.Parameter** is used to save any intermediatory values/filters to parameters, that can be accessed later requirement.

Pytorch functions used in the model

Cont...

7. **nn.Upsample** upsamples a given multi-channel 1D (temporal), 2D (spatial), or 3D (volumetric) data.
The input data is assumed to be of the form minibatch x channels x [optional depth] x [optional height] x width. Hence, for spatial inputs, we expect a 4D Tensor, and for volumetric inputs, we expect a 5D Tensor.

8. **nn.BCEWithLogitsLoss** This loss combines a Sigmoid layer and the BCELoss in one single class. This version is more numerically stable than using a plain Sigmoid followed by a BCELoss as, by combining the operations into one layer, we take advantage of the log-sum-exp trick for numerical stability.